



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/731,678	12/06/2000	Sung-Hee Do	A0734/7001 (EJR)	9300

7590 09/16/2003

Edward J. Russavage
Wolf, Greenfield & Sacks, P.C.
600 Atlantic Avenue
Boston, MA 02210

EXAMINER

VU, TUAN A

ART UNIT	PAPER NUMBER
----------	--------------

2124

DATE MAILED: 09/16/2003

6

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.		Applicant(s)	
	09/731,678		DO ET AL.	
	Examiner		Art Unit	
	Tuan A Vu		2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 06 December 2000.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-95 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1--95 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☒ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 16 April 2001 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
- 11) ☐ The proposed drawing correction filed on _____ is: a) ☐ approved b) ☐ disapproved by the Examiner.
If approved, corrected drawings are required in reply to this Office action.
- 12) ☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

- 13) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. _____.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
* See the attached detailed Office action for a list of the certified copies not received.
- 14) ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).
a) ☐ The translation of the foreign language provisional application has been received.
- 15) ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413) Paper No(s). _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. This action is responsive to the application filed December 6, 2000.
Claims 1-95 have been submitted for examination.

Specification

2. The disclosure is objected to because of the following informalities: there appears to be a misprint at "... of the design matrix an the at least one design parameter" (pg. 4, line 29), thereby rendering the sentence unclear. Appropriate correction is required.

Claim Objections

3. Claim 31 is objected to because of the following informalities: there appears to be a misprint at "... of the design matrix an the at least one design parameter" (pg. 84, line 20). The suggested correction is to replace the misprinted "an" with --- and --.
 4. Claim 14 is objected to because of the following informalities: there appears to be a missing list of elements after the reciting of "...selected from a group including:". The examiner has no way of interpreting such incompleteness, and will leave this claim from consideration in order to proceed on with examining the rest of the claims merits.
 5. Claim 15 is rejected to because it is practically a replica of claim 6 in terms of the limitations recited.
 6. Claim 52 objected to under 37 CFR 1.75(c), as being of improper dependent form for failing to further limit the subject matter of a previous claim. Applicant is required to cancel the claim(s), or amend the claim(s) to place the claim(s) in proper dependent form, or rewrite the claim(s) in independent form. Claim 52 recites exactly the limitations recited in the base claim
- 49.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

7. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

8. Claims 14, 15 and 16-30, 65 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 14 appears to be an incomplete sentence for no follow-up elements are recited to further specify the context enclosed with a “.”. The Examiner will not consider this claim until appropriate correction is performed.

Claim 15 is reciting the element “the software system” (pg. 82, line 24); but there is insufficient antecedent basis for this element. Examiner would interpret this as the software method as claimed.

Claims 16-30 are also rejected for being dependent upon claim 15.

Claim 65 is rejected because the recited element “...the design specification” (pg. 88, lines 29-30) is not provided with sufficient antecedent basis. Examiner would interpret this as the design matrix specification.

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

10. Claims 1 and 64 are rejected under 35 U.S.C. 103(a) as being unpatentable over Richburg, USPN: 5,159,687 (hereinafter Richburg) in view of Underwood, USPN: 6,523,027(hereinafter Underwood).

As per claim 1, Richburg discloses a digital information product comprising: a computer-readable medium and, stored therein, computer program instructions defining a software system that produces code based on a set of requirements and design specifications, or parameters by the programmer (e.g. col. 3, lines 23-54; col. 7, lines 37-50; *variables* – col. 5, lines 38-51; *knowledgeable, applications requirements* -- col. 10, line 41 to col. 11, line 24; col. 18, line 61 to col. 19, line 20; *Order Processing/user defined code* -Fig. 2; Fig. 3, 4; *specific variables or requirements* - col. 19, lines 3-9; *knowledgeable database, expert system code generator* -- col. 20, line 42 to col. 21, line 22).

But Richburg does not specify that the requirements are functional requirements but suggests accommodating target functionality when generating software code based on requirements database (e.g. *function of the knowledgeable* - col. 10, lines 41-48). The concept of requirement fulfilling by a software design is mostly known to mean establishing the functional aspect of the requirements with respect to the implemented code or design parameters. Underwood, in a method to generate code based on user or developer requirements inputs analogous to Richburg's method, discloses mapping functional requirements as specified by the system's users to code generated for validation (e.g. col. 84, lines 55-62; Fig. 27; col. 211, lines 35-41). In case Richburg does not specify that the requirements based on which to generate code are functional requirements, it would be obvious for a skill in the art at the time the invention was made to modify Richburg's method to map functional requirements as suggested above by

Underwood to the specified code design parameters or inputs by the developers as suggested by Richburg. The motivation would be obvious since code is generated to implement or fulfill business functionalities or a set of target functions hence functional requirements are first to be considered when validating with respect to design specifications.

As per claim 64, Richburg discloses a database format for designing a software system, comprising:

design identification information for identifying information describing the software system (e.g. *knowledgeable base*- Fig. 2; col. 7, lines 2-53; *Master Knowledgeable file* - col. 22, line 15 to col. 23, line 9); and

software code information associated with the software system (e.g. Fig. 2; col. 27, line 22 to col. 28, line 62 – Note: Knowledgeable files and ISF and ICF and/or header files data are equivalent to code generating information necessary to build target software system application).

But Richburg does not explicitly specify detailed design description information describing the structure and operating qualities of the software system even though Richburg discloses table information for describing program language structure of the code instructions (e.g. col. 25, line 5 to col. 27, line 11) and header files from the *knowledgeable* database (col. 27, line 22 to col. 28, line 62). Underwood, in a method to generate software to fulfil a requirements provided by a user/developer analogous to the method by Richburg, discloses project documentation, configuration legacy and baseline control (e.g. Fig. 21-22; Fig. 94; col. 77-79), and operating, source code environment and user interface database (e.g. Fig. 9-13, Fig. 26; col. 143-149) and metadata supporting dynamic input specifications (e.g. Fig. 15A), hence implicitly discloses detailed design information as claimed. It would have been obvious for one of ordinary

Art Unit: 2124

skill in the art at the time the invention was made to modify the information database to support the code processing as taught by Richburg so to add the detailed design information describing the software system structure and operating qualities as suggested by Underwood, because this would provide better informative data for further design improvement and/or reuse based on existing software operation information acquired from detailed description or metadata/legacy of previous code implementations.

11. Claims 2-13, 15-63, and 65-94 are rejected under 35 U.S.C. 103(a) as being unpatentable over Richburg, USPN: 5,159,687, and Underwood, USPN: 6,523,027, and further in view of Rudolph, "On a Mathematical Foundation of axiomatic design", 8/22/1996, ASME Design Engineering Technical Conference and Computers in Engineering Conference (hereinafter Rudolph).

As per claims 2-5, in reference to claim 1 above, Richburg combined with Underwood discloses mapping for correlating functional requirements to software design specifications or user's inputted implementation parameters, but does not specify program instructions to allow the developer to define a matrix describing relation between functional requirements (FR) and design parameters (DP) as recited in claim 2; nor does Richburg specify these instructions allow a programmer to manipulate into lower triangular form (re claim 3), to determine a decoupled design matrix (re claim 4), or to determine an uncoupled design matrix (re claim 5).

Correlating the functional requirements to the design logic, specifications or parameters has been a known concept in Software Engineering, and the use of data structure to map the FR and the DP are also common. Underwood, for example, discloses using a matrix to evaluate Firewall implementation against system requirements (e.g. col. 289, lines 1-13; Fig. 119). Rudolph, in a

Art Unit: 2124

method to use a mathematical approach in linking design objective to postulated requirements, discloses using a known concept which is a matrix to map FR to DP as claimed, to allow the developer to define such matrix to be manipulated in a lower triangular corner or decoupled form; to be defined in a uncoupled form as claimed (e.g. *Coupled design*, *Decoupled design*, *Uncoupled design* -- pgs. 2-4, ch. 2) in order to select the best design based on some axiomatic information. It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement to the mapping thus disclosed by Underwood/Richburg in establishing correlation between design and requirements so to add instructions code for generating a matrix structure using axiomatic approach as mentioned by Rudolph because this manipulation of the matrix into some desired patterns (e.g. coupled, uncoupled, or decoupled) for matching the requirements to the design resources would further enhance the optimization of resources while fulfilling the functional requirements as implemented by the design.

As per claim 6, Richburg discloses a method for producing software system, such method producing code based on a set of requirements and design specifications, or parameters (re claim 1 for rejection) but fails to specify explicitly that such set of requirements are functional requirements. But with the combination using Underwood's teaching, such limitation has been addressed as set forth in claim 1 above so that such method thereby comprises associating functional requirements (FR) of software and design parameters (DP). However, Richburg in combination with Underwood fails to disclose defining a design matrix for describing the relation between the FRs and the DPs and generating code based thereon.

The limitation of defining a matrix to correlate the software system FRs with the DPs in the generating of code has been addressed in claim 2 above, hence is rejected herein using the same rationale as set forth therein.

As per claim 7, Rudolph, as mentioned above in support to the combination with Richburg/Underwood, discloses a matrix where the diagonale elements are used, i.e. *Decoupled design* (e.g. pg. 3, 4th para.) just as recited by claim 4, and this would make this instant claim obvious using the rationale as set forth in claim 4 above.

As per claim 8, Richburg discloses using Case tools for processing of code structures into generating of code files (e.g. col. 5, line 52 to col. 6, 33). Likewise, Underwood utilizes Case tools to model the design objects used to implement the required specifications of the program logic (*Rational Rose* – Fig. 36, 122; col. 97, lines 24-62) and further describe the relationships between modules with entity-relation diagram (e.g. col. 96, lines 4-28) and activity of object interactions at runtime (e.g. Fig. 19, 20A), all of which being a representation of software objects or modules with interrelations equivalent to flow diagram of a software system. It would have been obvious for one of ordinary skill in the art at the time the invention was made to add to the Case tools of Richburg the case tools modeling and diagram flow representation as taught by Underwood in the process of generating code. The motivation is that the modeling with representation of system's components inter-dependency or flow is helpful in determining the relationships between code object or modules and thereby enabling an object-oriented design with all the benefits from the OOD modeling and modularization and reusability therefrom, as well as a better correlating of FR and DP derived from such modular representations.

As per claim 9-13, Underwood, by disclosing modeling via Case Tools such as Rational Rose, ERD, and Java ASP pages modeling (e.g. Fig. 18, 122, 47; col. 96, lines 4-28; col. 99 line 35 to col. 100, line 33) has implicitly disclose defining a software module corresponding to one functional requirement (re claim 9 – Note: each object in an Use case depict a functional requirement) ; defining a software module based upon a flow description (re claim 10); defining a plurality of software modules corresponding each to a FR (re claim 11); relating (re claim 12) the plurality of software modules by junctions (see Fig. 19, 122); and (re claim 13) combining such modules according to their junctions (Note: this combination of object modules linked by relationships is inherent to the modeling and code generation process in Rational Rose development). The motivation to modify Richburg's case tools so to implement the code using modeling with system diagram flow as taught by Underwood, and modularizing process thereby would be obvious herein using the rationale as set forth in claim 8 above.

As per claim 15, this claim can be rejected using the rejection of claim 6.

As per claims 16-19, these claims correspond to claims 3, 5, 7, and 4 respectively; and are rejected likewise, respectively.

As per claim 20, Richburg does not disclose generating a plurality of models and outputting of models for a code generator but discloses a plurality of files describing code specifications to be submitted to the code processor (e.g. Fig. 2-3). Using the teachings by Underwood in claim 8 (and again in claims 9-13) along with the Case Tools utilization by both Richburg and Underwood, this limitation would have been obvious for the same reasons as set forth therein.

As per claims 21-23, Richburg discloses case Tools and object reusability type of approach (e.g. *standardized software units, extensibility* - col. 3, line 49 to col. 4, line 16; col. 8, lines 1-17) and a programming language that is class-based (e.g. col. 12, lines 54-60) but does not explicitly specify generating (re claim 21) a diagram of a object-oriented structure, (re claim 22) a diagram in a UML format, or (re claim 23) a group of diagrams comprising use case and E-R diagram. But these limitations are implicitly disclosed by the teachings of Underwood as mentioned in claims 8-13 from above. These limitations would also been obvious in view of the rationale used therein, i.e. the motivation for modifying the class-based programming language and case tools and reusability approach by Richburg with the Rationale Rose case tools (i.e. object-oriented, UML format) as suggested by Underwood for modeling, or with E-R diagrams would have been obvious for the same reasons as mentioned in the rejections of claims 8-13.

As per claim 24, only Underwood discloses generating object-oriented entities (e.g. Fig. 18, 122, 47; col. 96, lines 4-28; col. 99 line 35 to col. 100, line 33; Fig. 54-56) in view of the teachings of using Object-Oriented case tools to model the business logic and generating code (re claim 8). The motivation to modify Richburg's teachings on case tools to implement the object-oriented modeling and diagramming for generating OO code would be obvious using the same rationale as used in claims 8-13 above.

As per claim 25, Richburg discloses case Tools and a programming language that is class-based (e.g. col. 12, lines 54-60) but it is Underwood who discloses defining OOP classes to fulfill a requirements (e.g. *implement ... requirements and design* - col. 13, line 10 to col. 14, line 26). The fact of using Rational Rose case tools to model class entities is equivalent to customizing requirements per object instances or class entity. The combination with Richburg's

case tools for generating code would be obvious for the same rationale mentioned in claims 21-23 above, and because of the benefits pertinent to developing in an object-oriented approach such as reusability, portability, modularization, extensibility, polymorphism, encapsulation, etc.

As per claims 26-27, since classes are known to be organized in a hierarchy of grand-parents, parents, and children, the limitations of defining a child, grand-child of a class using a particular FR would also been obvious by virtue of the above rejection.

As per claims 28-29, these claims would also been obvious for the same rationale used in claim 25-27 above.

As per claim 30, Richburg (in combination with Underwood/Rudolph) discloses generating source code (e.g. *Program file 16* – Fig. 2).

As per claim 31, Richburg discloses a method for designing software, such method comprising: defining a relation between a plurality of requirements of the software system and design parameters (e.g. col. 3, lines 23-54; col. 7, lines 37-50; col. 18, line 61 to col. 19, line 20; *variables* – col. 5, lines 38-51; *knowledgeable, applications requirements* -- col. 10, line 41 to col. 11, line 24; Fig. 3, 4; *specific variables or requirements* - col. 19, lines 3-9; *knowledgeable database, expert system code generator* -- col. 20, line 42 to col. 21, line 22).

But Richburg does not specify that the plural requirements are functional requirements; but this limitation has been addressed in claim 1 above using Underwood.

Nor does Richburg disclose explicitly that the software design involves object-oriented objects, nor does Richburg disclose representing at least one object-oriented object by at least one of the functional requirements (FR), and by at least one design parameters (DP). In view of Richburg's approach suggesting reusable of units of code (*standardized software units*,

Art Unit: 2124

extensibility - col. 3, line 49 to col. 4, line 16), such reminiscent of object-oriented extensibility and modularization propriety (e.g. col. 2, lines 26-37) as afore-mentioned in claim 21, it would have been obvious for one of ordinary skill in the art at the time the invention was made to modify Richburg's approach of reusing code units so to include the modeling approach by Underwood as mentioned in claim 8, 21 above so to represent at least one FR or DP with the object-oriented object as taught by Underwood, e.g. via Rationale Case Tools (see e.g. Underwood: Fig. 18, 122, 47; col. 96, lines 4-28; col. 99 line 35 to col. 100, line 33) because of the benefits of object-oriented such as suggested by Richburg like enabling extensibility and modularization; and of the reasons as mentioned in the rejection of claim 8, 21 above.

Nor does Richburg teach defining a design matrix to correlate a plurality of FRs and DPs. This limitation has been addressed in claims 2-4 above; hence is rejected herein likewise.

Nor does Richburg disclose representing a method of at least one object-oriented software object by a product of a portion of the matrix and at least on DP. But in view of the mathematical approach by Rudolph using a matrix multiplication scheme to derive an matrix association or product element (or method as claimed) by matching FR and DR matrices (e.g. pg. 2, ch. 2: math expressions (1) and (2)) as used in claims 2-4 rejection, this limitation would also been obvious for the same rationale.

As per claim 32, Richburg in combination with Underwood disclose defining in a database functional requirements and system constraints (re claim 31).

As per claims 33-35, only Rudolph teaches the matrix approach to equate a set of FRs with a product by or association with set of DPs (see Rudolph: pg. 2-4, ch.2); the organizing of FRs into a leaf elements, or DPs into leaf elements (rows or columns) of the matrix, or process

Art Unit: 2124

variable into a leaf would have been thereby suggested, the process variable (PV) or design parameters being interchangeable sets to be mapped against the FR set in order to correlate program specification to FR as suggested by Richburg (e.g. *variables* – col. 5, lines 38-51; *knowledgeable statements, parameters substitution* - col. 10, line 47 to col. 11, line 62). As has been obvious according to the corresponding rejection of claim 31, the usage of the matrix to determine correlation or method between program variables or design parameters as taught by Richburg and further enhanced by Underwood and Rudolph, the limitation to set FR and DP or PV as leaf elements in the row or columns of the matrix as suggested by Rudolph would also been obvious for the same rationale used in claim 31 above.

As per claim 36, Richburg discloses a software designing method comprising: defining a software system by defining a relation between FRs of the system and DPs implementing the system (re claim 31 for corresponding rejection); but does not specify defining an object-oriented object. This last limitation has been addressed in claim 31 above using Underwood's teachings.

Nor does Richburg (combined with Underwood) disclose defining a design matrix for correlating the FRs and the DPs; nor does Richburg teach defining a method that may define on a OO object, such OO object and such method are related to the design matrix. These limitations have also been addressed in claim 31 above.

As per claim 37-40, both Richburg and Underwood disclose mapping of requirements against OO program specifications or parameters but do not teach such mapping by using a design matrix to map FRs and DPs (such limitation been addressed above using Rudolph); nor does Richburg (with Underwood's teachings) disclose object class, its instance, its behavior

Art Unit: 2124

against 1st level, 2nd level, or 3rd level of FR, respectively and all those levels against a hierarchy of FRs (re claim 37, 38, 39, 40) . Rudolph further discloses such mapping being applied to hierarchy of objects in accordance with nth level rank matrix mapping (e.g. pg. 4, Fig. 3; pg. 4-5, ch. 3) for evaluation of the most optimal design choice as earlier suggested by Underwood (col. 289, lines 1-13; Fig. 119) in claim 2 above. It would have been obvious for one of ordinary skill in the art at the time the invention was made to modify the OO approach by Richburg (combined with Underwood) to map FR and program objects so that a hierarchy of objects, such as class, class instance or class behavior are matched via the matrix of FRs organized into multi-rank levels as suggested by the design evaluation by Rudolph. The motivation would be obvious because this would enable Richburg's method to address each and every level of hierarchy of objects making up the object-oriented approach code implementation suggested by Richburg and furthered by Underwood, thereby enabling a more accurate validation of every aspects of the OO assembling of elements in accordance to the level of subdivision imparted to the corresponding FRs as mentioned by Rudolph's matrix-based evaluation.

As per claim 41, in view of the teachings by Rudolph to use design matrix and the combination as mentioned in claim 36, the mapping of FR into the implementation domain via using DPs would also been obvious for the same grounds as set forth therein.

As per claims 42-43, refer to claims 7 and 5, respectively.

As per claims 44-47, these claims correspond to claims 20, 21, 23 and 22 respectively, hence are rejected using the corresponding rejections therein.

As per claim 48, Richburg discloses a database format for designing a software system, comprising: a software design specifications wherein is defined a relation between a plurality of

requirements and design parameters (e.g. col. 3, lines 23-54; col. 7, lines 37-50; col. 18, line 61 to col. 19, line 20; *variables* – col. 5, lines 38-51; *knowledgeable, applications requirements* -- col. 10, line 41 to col. 11, line 24; Fig. 3, 4; *specific variables or requirements* - col. 19, lines 3-9; *knowledgeable database, expert system code generator* -- col. 20, line 42 to col. 21, line 22) and a software code produced by the design specifications (Fig. 2; Fig. 3, 4 – Note: the storing of software code for future reuse or testing purposes in a database is inherent).

But Richburg does not explicitly specify that such plurality of requirements is one of functional requirements. This limitation has been addressed in claim 1 using Underwood.

Nor does Richburg disclose defining a design matrix to describe a relation between the FRs and the DPs. This limitation has also been addressed in claim 2 above.

Nor does Richburg disclose design specification for at least one software object being represented by at least one or more FRs, or specification for data used by such object being represented by at least one or more DPs. But this limitation would have been obvious in view of the object-oriented teachings by the combination Richburg/Underwood as being addressed in claims 37-41.

As per claim 49, Richburg discloses identifying the software code in the database (e.g. col. 7, lines 2-53; Fig. 2).

As per claim 50, Richburg discloses specifications stored as knowledgeable files in a database to correspond with the requirements database (re claim 1) but does not disclose storing therein of detailed design description including the design matrix. But in view of the combination using Rudolph to implement a matrix to correlate Richburg FR and DP (with Underwood's teachings) as seen in claim 2, this limitation would also been obvious. One of

ordinary skill in the art would be motivated to provide database storage of design description including such matrix implementing such as taught by Rudolph for the same reason to support the mapping of Richburg's system just as set forth in claim 2 above, i.e. this manipulation of the matrix into some desired patterns (e.g. coupled, uncoupled, or decoupled) for matching the requirements to the design resources would further enhance the optimization of resources while fulfilling the functional requirements as implemented by the design.

As per claim 51, this claim would also have been obvious for the rationale to include a design matrix as set forth in claim 50 into the database storing the FR and the DPs as taught by Richburg would also be used herein.

As per claim 52, refer to claim 49.

As per claim 53, this claim corresponds to the organization of software design parameters or objects specifications into levels or hierarchy analogous to the level of mapping of object-oriented objects to level of FRs as discussed in claims 37-40 above; hence is rejected herein using the rationale as set forth therein. As for the referencing of such plurality of levels of DPs by another software system as recited in the instant claim, the object-oriented approach as addressed using Underwood to enhance Richburg as set forth in claims 36-40, would have implicitly disclose this limitation, given the hierarchy of class organization and method calling structure inherent to an OO programming design.

As per claim 54, see Richburg (Fig. 1-3; col. 3, lines 49-65; col. 7, lines 37-53).

As per claim 55, see Richburg discloses user interface and database, and searching therein to obtain elements to create a new software system (e.g. col. 3, lines 49-65; Fig. 2a-10; col. 6, lines 34-48).

As per claim 56, Richburg discloses the interaction of new software and several elements of the software system (e.g. col. 18, line 61 to col. 19, line 34) but Richburg does not disclose verifying if the interface is operable. Underwood, in a analogous method using user interface to specify the creation of a new software system disclose the checking consistency of data gathered during a session of user interfacing to support the fulfillment of a service request for application code (e.g. Fig. 15B, 18A, 20, 153-154). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the user interface of Richburg so to incorporate the correctness checking of the activities in the user interface during the creation of the new software system code as suggested by Underwood because this would enhance the optimization of resources usage by obviating potential failures due to interface data retrieval incompatibilities or user-system communications data inconsistency problems as mentioned by Underwood.

As per claim 57, Richburg mentions about consistency of database to support a large scale of applications and user interfacing (e.g. col. 8, lines 1-17) and control checking for correctness in program generating based on the information of the database (e.g. col. 23, line 61 to col. 24, line 23); hence has disclosed consistency of design.

As per claim 58, see Richburg (e.g. col. 18, line 61 to col. 19, line 20; col. 25, line 4 to col. 27, line 11 – Note: operands and mnemonics --or keywords--, and comments of SEL language stored in knowledgeable files and Application database are equivalent to design parameter information stored in database).

As per claim 59, Richburg discloses an executable form (e.g. col. 30, lines 57-62 – Note: assembly language is implicitly disclosing that source code is executable by a machine level).

As per claim 60, Richburg discloses source code (e.g. Fig. 2).

As per claim 61, Richburg disclose control checking of code constructs against knowledgeable in the database (re claim 57) and mapping of requirements with code specifications via a code processing unit (Fig. 2-5); but does not specify consistency checking of implemented design between some level of DP. But in view of the checking steps as shown in the Fig. 2-4, from high module level to low level instruction constructs of an implemented form of the program specifications, this limitation is disclosed.

As per claim 62, Richburg suggested reusable framework (e.g. col. 24, line 38 to col. 25, line 2) and Underwood teaches object-oriented design (re claims 21-24). But Richburg does not specify consistency checking of level of design so that the second level is child of first level, even though Richburg teaches about class in implementation of code (col. 12, lines 54-60). In view of the rationale to combine Underwood's approach to use object-oriented approach to implement the requirements mapping and code generation of Richburg as set forth in claim 21-24 above, it would have been obvious for one of ordinary skill in the art at the time the invention was made to modify the code checking as suggested by Richburg from above so that it does consistency checking between hierarchy of classes, e.g. child-parent checking, as suggested by Underwood from above because of the same motivations as mentioned in claims 21-24.

As per claim 63, the rationale used in claim 56 is herein used to address this instant claim.

As per claim 65, in reference to claim 64, Richburg discloses a detailed design description but fails to define a design matrix to establish relation between FRs and DPs. But this limitation has been addressed in claim 2 above. Nor does Richburg disclose design

specification for at least one software object being represented by at least one or more FRs, or specification for data used by such object being represented by at least one or more DPs. But this limitation would have been obvious in view of the object-oriented teachings by the combination Richburg/Underwood as being addressed in claims 37-41.

As per claim 66, see Richburg (e.g. Fig. 2; col. 27, line 22 to col. 28, line 62).

As per claim 67, this claim corresponds to claim 15, hence is rejected using the rationale as set forth therein.

As per claim 68, refer to claim 32.

As per claims 69-71, refer to corresponding rejections of claims 33-35, respectively.

As per claims 72 and 74, refer to rejections of claims 22 or 47.

As per claim 73, see claim 59.

As per claim 75, Richburg does not disclose diagram describing a software code, but according to the modeling approach by Underwood as set forth in the rejection of claims 21-24, the limitation such as to describe software with a diagram such as a class diagram, interaction diagram, collaboration diagram, sequence diagram, state diagram, activity diagram as claimed would have been implicitly disclosed by the modeling with Case tools (e.g. Rational Rose; Fig. 18, 122, 47; col. 96, lines 4-28; col. 99 line 35 to col. 100, line 33; Fig. 54-56) as taught by Underwood; and would have been obvious for the same reasons used in claims 21-24.

As per claim 76, the limitation to associate at least one object/element of the diagram modeling by Underwood to the design matrix would have been obvious by virtue of the rationale using Underwood's validation of requirements (e.g. col. 289, lines 1-13; Fig. 119) and Richburg's relating of requirements to software specification as set forth in claim 2.

As per claim 77, use-case is implicitly disclosed in Underwood's teachings from claim 75, hence the limitation of using use-case based on the design matrix would have been obvious by virtue of the above rationale from claim 76.

As per claim 78, Richburg, combined with Underwood, discloses validating user's needs or requirements to conform to requirements but does not disclose hierarchy of user's needs associated with layers of use-case diagram. By virtue of the rationale as set forth in claims 37-41, using the hierarchy and multi-level structuring of the matrix elements by Rudolph and the modeling and user's requirements mapping by Underwood/Richburg, this above limitation would also be obvious using the same grounds as set forth in claims 37-41, because organizing objects or elements of an use-case is equivalent to organizing object-oriented elements by Underwood in the mapping against the multi-levels of requirements as taught Rudolph's matrix, especially when use-case is fundamentally a form of modeling instance for customizing or fulfilling a user's requirements.

As per claim 79, since the layers such as relation layer, use case layer, and actor layer are all implicitly disclosed in the case tools by Underwood, this claim is rejected with the same rationale as set forth in claim 77.

As per claim 80-82, since actor layer representing more than one actor is implicit to the Rational Rose taught by Underwood, and that a use case describing a user needs is inherent in such Case tools, or that a relation layer describes a relation between customer's needs, these claims would have been obvious with the same rationale used in claim 77.

As per claims 83-85, these claims are implicitly disclosed because most modeling sets of data are stored in some files in the development tools such as exemplified by the Rational Rose

Art Unit: 2124

Case tools as taught by Underwood (e.g. Object Modeling -col. 97-99). In combination with the rationale as used in claims 21-23, or claims 76-77, these claims are herein rejected for being obvious over the combination and rationale as set forth therein.

As per claim 86, Richburg discloses a computer product with a program to performs a method for rendering an user interface, such method comprising displaying a software design interface (e.g. col. 4, lines 32-41; *edit pull-down menu* - Fig. 2; col. 5, line 60 to col. 7, line 16) upon which a software design is based and described in terms of correlating a set of requirements and design parameters (DP) implementing the software design (e.g. col. 3, lines 23-54; col. 7, lines 37-50; *variables* – col. 5, lines 38-51; *knowledgeable, applications requirements* -- col. 10, line 41 to col. 11, line 24; col. 18, line 61 to col. 19, line 20; *Order Processing/user defined code* -Fig. 2; Fig. 3, 4; *specific variables or requirements* - col. 19, lines 3-9; *knowledgeable database, expert system code generator* -- col. 20, line 42 to col. 21, line 22).

But Richburg does not specify that the requirements associated with the DPs are functional requirements (FR). But this limitation has been addressed in claim 1 using Underwood's teachings.

Nor does Richburg disclose displaying a set of interface requirements upon which the software design is based, nor does Richburg disclose displaying a design matrix for correlating said FRs to the DPs. In view of the teachings by Rudolph for generating a matrix describing a relation between the FRs and the DPs as mentioned in claim 2 above, it would have been obvious for one of ordinary skill in the art at the time the invention was made to add to the user interface as suggested by Richburg so that the matrix as taught by Rudolph is displayed, describing thereby the relation between the database stored requirements and the code specifications as

indicated by the user. The motivation is that the mapping to correlate user's requirements to the design specifications as intended by Rudolph in order to generating the application code would be enhanced by providing a dynamic information conveying via graphical visualizing of the mathematical approach as suggested by Rudolph, thus alleviating potential errors that would otherwise incur with no such matrix display when developing the code.

As per claim 87, using the teachings of Rudolph as used in claims 2-5 (e.g. *Coupled design*, *Decoupled design*, *Uncoupled design* -- pgs. 2-4, ch. 2), the limitations of a decoupled, uncoupled, or coupled design would have been obvious in view to such rationale set forth therein.

As per claims 88-89, these claims would have been obvious in view of the combination as set forth in the rejection of claim 2-5 or claim 87 from above.

As per claims 90-91, the teachings by Rudolph disclose defining or indicating an accuracy of design with respect to an optimum design (e.g. *best design* - pg. 3, right column, 1st para.) ; or defining/indicating a range of acceptable inputs (e.g. *upper and lower limits* - pg. 5, ch. 3.1) operable by the design matrix are disclosed, rendering the instant limitations obvious according to the rationale of claims 2-5 or claim 87 above.

As per claims 92-93, these claims would also been obvious in view of the rationale in rejections of claims 37-41.

As per claim 94, by choosing the best design using the matrix mathematics method, Rudolph discloses the robustness of the design choice; hence the combination using Rudolph, and Richburg/Underwood as set forth in claims 2-5 (and claim 87) would also render the instant limitation obvious for the same rationale therein.

12. Claim 95 is rejected under 35 U.S.C. 103(a) as being unpatentable over Richburg, USPN: 5,159,687, and Underwood, USPN: 6,523,027, in view of Rudolph, "On a Mathematical Foundation of axiomatic design", 1996; as applied to claim 92, and further in view of Crampton et al., USPN: 6,415,196 (hereinafter Crampton).

As per claim 95, Richburg discloses rendering of a software code based on requirements association with database of *knowledgeable* and specifications provided by users; but does not teach arrangement of design parameters into a Gantt chart depicting production dependencies of the software design. Underwood, in an analogous system to fulfill user's requirements or specifications (re claim 2), discloses delivery and versioning of production code release and development management (e.g. *project management* – Fig. 43; production steps 2504. 2510 -- Fig. 25; *PVCS* –, 30-31; col. 89, line 16 to col. 90, line 32). Further, Crampton, in a method to manage the development process using modeling and requirement fulfilling (see Crampton: Fig. 7, 10) like Richburg's method and analogous to the management of software development by Underwood, discloses the use of Gantt's charts to depict production dependencies and evolution of software entities or resources over time (e.g. col. 31, lines 37-63). It would have been obvious for one of ordinary skill in the art at the time the invention was made to add to the software production and delivery management as suggested by Underwood as in the combination Richburg/Underwood from claim 92 from above the implementation using Gantt chart as taught by Crampton, because this graphical depiction of a timeline evolution of resources would enhance the planning and scheduling of resources when producing the software deliverables as intended so as to support more efficiently the use of system resources in a life cycle of the program development.

Conclusion

12. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Tracz et al., "Domain-Specific Software Architecture Engineering Process Guidelines", 1993, ADAGE-IBM-92-02, Version 2.0, disclosing junctions between software objects created from requirements association.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (703)305-7207. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks
Washington, D.C. 20231

or faxed to:

(703) 872-9306 (for formal communications intended for entry)

or: (703) 746-8734 (for informal or draft communications, please label
"PROPOSED" or "DRAFT")

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive,
Arlington. VA. , 22202. 4th Floor(Receptionist).

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

Application/Control Number: 09/731,678

Page 25

Art Unit: 2124

VAT

September 7, 2003

Kakali Chaki

**KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100**